Python Cheat Sheet: The Basics

run of deal proposed block run of deal proposed block run of deal proposed block restrictly with all uppercase letters restrictly with all uppercase of the attrop inside the restrictly with all uppercase of the attrop inside the restrictly with all uppercase of the attrop inside the restrictly with all uppercase of the attrop inside the restrictly with all uppercase of the attrop inside the restrictly with all uppercase of the attrop inside the restrictly with all uppercase of the attrop inside the all uppercase of the attrop inside the all uppercase of the attrop inside the all uppercase of attrop inside the all uppercase of attrop inside the all uppercase of the attrop inside the all uppercase of attrop inside the all uppercase of attrop inside the annotation of	Python Data Types String	List Changeable collection of objects $m_{X} collection = [1,\ 1,\ 3.12,\ False,\ "Hi"]$	Indexing Accessing data from a string, list, or tuple using an element number my_string[element_number]
# returns the length of a list serions serions serions the length of a list serions serions serions serions serions serions serions serions serions serions serions serions serions serions serions serions serions serions serions serions serio	inacters or data stored as text a = "He]]0"		mstringletement_number] my_collection[element_number] my_tup[element_number]
# Add multiple tiens to a list # Add a single tien to a list # Clone a list # Conclettion, 2 = "", "", "", "", # Clone a list # Conclettion, 2 = "", "", "", "" # Clone a list # Conclettion, 2 = "", "", "", "" # Clone a list # Conclettion, 2 = "", "", "", "" # Conclettion # Returns to the set # Returns to the set # Returns to the set # Returns tien to the set # Returns to the set # Returns tien tien to the set # Returns to the set a mid b # Returns the set a subset of b, false otherwise # Returns the union of set a and b # Returns the union	String Operations		Slicing Accessing a subset of data from a string, list, or tuple using element numbers from start to stop -1
# Add a stuple teem to a list # Collection.appendivising to a list and a secret the object of a list at a specified index ## Collection appendivision and a specified index ## Collection 2	# returns the string with all uppercase letters my_string.upper()	# Add multiple items to a list my_collection.extend(["More", "Items"])	my_string[start:stop] m_collection[start:stop]
for the first instance of the string inside the # Delete the object of a list at a specified index # Cline a list # Concatenate two lists # Concatenate two lists # Concatenate two lists # Concatenate two lists of finate of first of finate # Concatenate two lists of finits of finate # Concatenate two lists of finits of finate # Concatenate two lists of finits of finate # Concatenate two lists of the second in my_string # Concatenate two lists of finits of finate # Concatenate two lists of finits of finits of finits # Concatenate two lists # Convert a list to a list to a list to a set # Add an item to the set # Add an item to the set # Returns from a set a and b # Returns from the unit of set a and b # Returns from the unit of set a and b # Returns from the unit of set a and b # Returns from the unit of set a list of a subset of b, false otherwise # Set Description of a subset of b, false otherwise	# returns the length of a string len(my_string)	# Add a single item to a list my_collection.append("Single")	my_tuptstart:stopj Comparison Operators
## Clone a list ## Conceinate two lists ## Conceinate two lists or floats ## Conceination = 11,2,3,4,5] ## Conceination = 11,2,3,4,5] ## Conceination = 11,2,3,4,5] ## Add an item of the is not in a list, returns Boolean ## Conceination = 11,2,3,4,5] ## Add an item of the is not in a list, returns Boolean ## Add an item of the is not in a list, returns Boolean ## Add an item of the set ## Add an item to the set ## Add an item to the set ## Add an item to the set ## Returns the union of set a and b ## Returns the union of set a and b ## Returns The union of set a and b ## Returns The union of set a and b ## Returns The union of set a subset of b, false otherwise ## Add an item of the is a subset of b, false otherwise ## Returns The in the interpretation is a subset of b, false otherwise ## Returns The interpretation is a subset of b, false otherwise ## Returns The interpretation is a subset of b, false otherwise ## Returns The interpretation is a subset of b, false otherwise ## Add an item interpretation is a subset of b, false otherwise ## Returns The interpretation is a subset of b, false otherwise ## Add an item interpretation	# returns the index of the first instance of the string inside the # subject string, otherwise -1 $\rm my_string.find('l')$	$\#$ Delete the object of a list at a specified index $\mbox{del}(m_J.collection[2])$	Comparison Operators compare operands and return a result of true or false Equal
## Concatenate two lists ## Concatenate two lists ## Concatenate two lists ## Calculate the sum of a list of ints or floats ## Calculate the sum of a list of ints or floats ## Calculate the sum of a list of ints or floats ## Calculate the sum of a list of ints or floats ## Calculate the sum of a list of ints or floats ## Calculate the sum of a list of ints or floats ## Calculate the sum of a list, returns Boolean ## Check if an item is not in a list, returns Boolean ## Check if an item is not in a list, returns Boolean ## Connect a list is a list ## Returns the set a minus b ## Returns supersection of set a and b ## Returns theresection of set a and b ## Returns True if a is a subset of b, false otherwise ## Returns True if a is a subset of b, false otherwise ## Returns True if a is a subset of b, false otherwise ## Returns True if a is a subset of b, false otherwise ## Returns True if a is a subset of b, false otherwise ## Returns True if a is a subset of b, false otherwise ## Returns True if a is a subset of b, false otherwise ## Returns True if a is a subset of b, false otherwise ## Returns True if a is a subset of b, false otherwise ## Returns True if a is a subset of b, false otherwise	# replaces any instance of the first string with the second in $m_{\rm J}$ string $m_{\rm J}$ string.replace('H', 'C')	# Clone a list clone = my_collection[:]	a == b Less Than
# Calculate the sum of a list of ints or floats number_collection = [1,2,3,4.5] sum(number_collection) # Check tf an item is not in a list, returns Boolean item in my_collection # Check tf an item is not in a list, returns Boolean item not tan my_collection # Check tf an item is not in a list, returns Boolean # Check tf an item is not in a list, returns Boolean # Check tf an item is not in a list, returns Boolean # Check tf an item is not in a list, returns Boolean # Check tf an item is not in a list, returns Boolean # Check tf an item is not in a list, returns Boolean # Convert a list to a set # Returns as a list # Returns set a minus b # Returns set a minus b # Returns the union of set a and b # Returns True if a is a subset of b, false otherwise # Returns True if a is a subset of b, false otherwise # Returns True if a is a subset of b, false otherwise # Returns True if a is a subset of b, false otherwise	Integer A whole number	<pre># Concatenate two lists my_collection_2 = ["a", "b", "c"] my_collection_3 = my_collection + my_collection_2</pre>	a < b Greater Than
# Check if an item is in a list, returns Boolean # them it mm_collection # them it mm_collection # them not in my_collection Set Unodered collection Set Unodered collection \$	my_integer = 12321	<pre># Calculate the sum of a list of ints or floats number_collection = [1,2,3,4.5]</pre>	a > b Greater Than or Equal
# Check if an item is in a list, returns Boolean item in my_collection # Check if an item is not in a list, returns Boolean # Check if an item is not in a list, returns Boolean # Check if an item is not in a list, returns Boolean Set	umber	sum(number_collection)	a v m
# Check if an item is not in a list, returns Boolean tem not in my_collection Set Unodered collection :: 1, 12: 'laptop', (0.0):'center') # Convert a list to a set # Add an item from a set # Add an item from a set # Returns set a minus b # Returns set a minus b # Returns set and b # Returns the union of set a and b # Returns the union of set a and b # Returns the union of set a and b # Returns the union (b) # Returns the union of set a and b # Returns the union (b) # Returns the union (c) # Returns		# Check if an item is in a list, returns Boolean item in my collection	Less Than or Equal
Set Unodered collection of unique objects a = {1000, 3.12, False, "Bye"} b = {1000, 3.12, False, "Bye"} Set Operations # Convert a list to a set my_set = set([1,1,2,3]) # Add an item to the set a.add(4) # Remove an item from a set a.emove("Bye") # Returns set a minus b a.difference(b) # Returns intersection of set a and b a.intersection(b) # Returns the union of set a and b a.union(b) # Returns True if a is a subset of b, false otherwise a.ssubsetel of b, false otherwise		# Check if an item is not in a list, returns Boolean item not in my_collection	a <= b Non Forum
Deficiency as a list ctionary as a list Returns the union of set a and b Returns the union of set a and b Returns True if a is a subset of b, false otherwise Returns True if a is a subset of b, false otherwise Returns True if a is a subset of b, false otherwise Set Operations # Returns the union of set a and b # Returns True if a is a subset of b, false otherwise # Returns True if a is a subset of b, false otherwise # Returns True if a is a subset of b, false otherwise	Discrete value true or false		a != b
pairs ': 1, 12: 'laptop', (0,0):'center') Set Operations # Convert a list to a set # Convert a list to a set # Add an item to the set a.add(4) # Remove an item from a set a.add(4) # Returns set a minus b a.difference(b) # Returns intersection of set a and b a.intersection(b) # Returns True if a is a subset of b, false otherwise a.iscubsec(b) # Returns True if a is a subset of b, false otherwise a.iscubsec(b) # Returns True if a is a subset of b, false otherwise	30	Set Unordered collection of unique objects	
Set Operations # Convert a list to a set my_set = set([1,1,2,3]) # Add an item to the set a.add(4) # Remove an item from a set a.emove("Bye") # Returns set a minus b a.difference(b) # Returns intersection of set a and b a.intersection(b) # Returns True if a is a subset of b, false otherwise a.isoubset(b) # Returns True if a is a subset of b, false otherwise		a = {100, 3.12, False, "Bye"} b = {100, 3.12, "Welcome"}	Python Operators
# Convert a list to a set my_set = set([1,1,2,3]) # Add an item to the set a.add(4) # Remove an item from a set a.emove("Bye") # Returns set a minus b a.difference(b) # Returns intersection of set a and b a.intersection(b) # Returns the union of set a and b a.union(b) # Returns True if a is a subset of b, false otherwise a.issubset(b) # Returns True if a is a subset of b, false otherwise	collection of key-value pairs onary = {'banana': 1, 12: 'laptop', (0,0):'center'}	Set Operations	· +: Addition
# Add an item to the set a.add(4) # Remove an item from a set a.remove("Bye") # Returns set a minus b a.difference(b) # Returns intersection of set a and b a.intersection(b) # Returns the union of set a and b a.union(b) # Returns True if a is a subset of b, false otherwise a.issubset(b) # Returns True if a is a subset of b, false otherwise	/ Operations	<pre># Convert a list to a set my_set = set([1,1,2,3])</pre>	- : Multiplication - /: division - /: division (Result rounded to the nearest integer)
# Remove an item from a set a.remove("Bye") # Returns set a minus b a.difference(b) # Returns intersection of set a and b a.intersection(b) # Returns the union of set a and b a.union(b) # Returns True if a is a subset of b, false otherwise a.isubset(b) # Returns True if a is a subset of b, false otherwise	s value using key ionary['banana']		Conditional Operators
# Returns set a minus b a.difference(b) # Returns intersection of set a and b a.intersection(b) # Returns the union of set a and b a.union(b) # Returns True if a is a subset of b, false otherwise a.issubset(b)		# Remove an item from a set a.remove("Bye")	Conditional Operators evaluate the operands and produce a true of false result
# Returns intersection of set a and b a.intersection(b) # Returns the union of set a and b a.union(b) # Returns True if a is a subset of b, false otherwise a.issubset(b)		# Returns set a minus b a.diference(b)	And - returns true if both statement a and b are true, otherwise false
# Returns the union of set a and b a.union(b) # Returns True if a is a subset of b, false otherwise a.issubset(b)		a and	or and or
# Returns True if a is a subset of b, false otherwise a.tsubset(b)	ale collection of objects	union of set a and	a or b Not - returns the opposite of the statement
	, 3.12, False, "Ht")		not a



Returns True if a is a superset of b, false otherwise a.issuperset(b)



SKILLS NETWORK 💽

Loops

For Loops

Executes loop x number of times for x in range(x):

for x in iterable:

```
# Executes loop for each object in an iterable like a string, tuple,
                                                   list, or set
```

While Loops

```
while statement:
# Executes the loop while statement is true
```

Conditional Statements

```
elif statement_2: \# \ Execute \ if \ statement_1 \ is \ false \ and \ statement_2 \ is \ true
                                                                                                                                                                                       # Execute if all previous statements are false
                                     # Execute of statement_1 is true
if statement_1:
```

Try/Except

```
except:
# Code to execute if there is any exception that has not been handled
                                                              # Code to execute if there is an error of type a
                                                                                                                             # Code to execute if there is an error of type b
                                                                                                                                                                                                                                                              # Code to execute if there is no exception
# Code to try to execute
                                                                                                    except b:
                                  except a:
                                                                                                                                                                                                                           else:
```

Error Types

- · IndexError When an index is out of range
- · NameError When a variable name is not found
- SyntaxError When there is an error with how the code is written
 - ZeroDivisionError When your code tries to divide by zero

Range

Produce an iterable sequence from 0 to stop-1 range(stop)

```
Produce an interable sequence from start to stop-1 incrementing by step
                                                                                      range(start, stop, step)
```

Webscraping

Python Cheat Sheet: The Basics

```
# Find the first instance of an HTML tag
soup.find(tag)
                                                                                             # Parse HTML stored as a string
soup = BeautifulSoup(html, 'html5lib')
# Import BeautifulSoup
from bs4 import BeautifulSoup
                                                                                                                                                                                           # Returns formatted html
soup.prettify()
```

Requests

Find all instances of an HTML tag soup.find_all(tag)

```
# Send a get requests to the url with optional parameters
                                                                                                                                                                                      response = requests.get(url, parameters)
# Import the requests library
                                        import requests
```

```
# Get the status code of the response
                                                                                                                              # Get the headers of the request
                                                                                                                                                                                               # Get the body of the requests
# Get the url of the response
                                                                                                                                                              response.request.headers
                                                                                                response.status_code
                                      response.url
```

Get the content of the response in text # Get the content of the response in json response.headers response.json() response.text

Get the headers of the response

response.request.body

Send a post requests to the url with optional parameters requests.post(url, parameters)

Functions

```
def function_name(optional_parameter_1, optional_prameter_2):
                                                                                                                                          return optional_output
# Create a function
```

output = function_name(parameter_1, parameter_2)

Calling a function

Working with Files

Reading a File

```
# Reads a certain number of characters of a file
                                                                                                              # Returns the mode the file was opened in
                                                                                                                                                                                                               # Reads the contents of a file
# Opens a file in read mode
                            file = open(file_name, "r")
# Returns the file name
                                                                                                                                                                                                                                                                                                                                          file.read(characters)
                                                                                                                                                                                                                                           file.read()
                                                                                    file.name
                                                                                                                                            file.mode
```

Read a single line of a file file.readline() # Read all the lines of a file and stores it in a list file.readlines() # Closes a file file.close()

Writing to a File

```
# Opens a file in write mode
                                 file = open(file_name, "w")
```

Adds content to the end of a file # Writes content to a file file.append(content) file.write(content)

Objects and Classes

```
def __init__(self. optional_parameter_1, optional_parameter_2):
                                                                                                     self.attribute_1 = optional_parameter_1
                                                                                                                                        self.attribute_2 = optional_parameter_2
                                                                                                                                                                                                       method_name(self, optional_parameter_1):
                                                                                                                                                                                                                                                                                                                                                                                                  object = class_name(parameter_1, parameter_2)
                                                                                                                                                                                                                                                                                return optional_output
                                                                                                                                                                                                                                                                                                                                                               # Create an instance of a class
                                                                                                                                                                                                                                             # Code to execute
# Creating a class
                                  class class_name:
                                                                                                                                                                                                             def
```

object.method_name(parameter_3)

Calling an object method